Yale University

Policy Optimization: Policy Gradients and PPO

S&DS 265 · Introductory Machine Learning · Lecture 21

Zhuoran Yang

Fall 2026

AI usage: figures were generated, and these slides polished, with the help of AI tools.

Outline

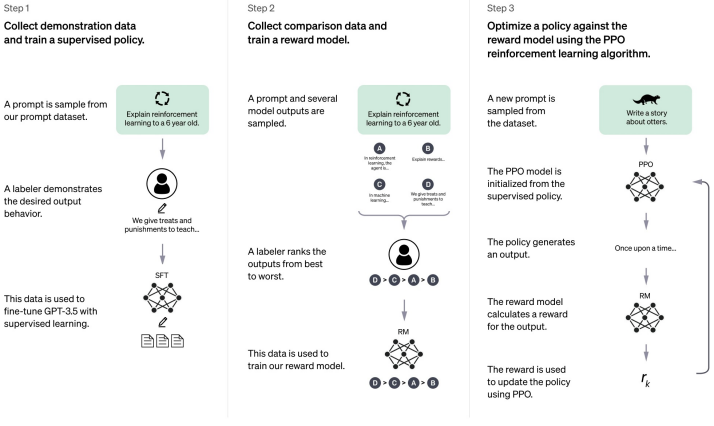
Learning Objectives

In this lecture you will learn:

1. How replacing the bandit reward by the return gives REINFORCE, and why its variance is large;
2. How the performance difference lemma is an exact finite difference, with Q^π as the gradient;
3. How the policy gradient theorem is that identity in the limit of a small step;
4. How an actor–critic algorithm estimates advantages with a least-squares critic and a frozen target;
5. How the PPO objective follows once the occupancy distribution is frozen, the ratio is reweighted, and the step is clipped.

Outline

Motivating Success: How ChatGPT Was Trained



Demonstrations, then a reward model, then step 3: reinforcement learning — and the algorithm named there is **PPO**.

Ouyang et al. (2022), OpenAI; diagram as published, credited educational use.

The Algorithm in Step 3

Proximal Policy Optimization Algorithms

Schulman, Wolski, Dhariwal, Radford, Klimov

arXiv:1707.06347 · 2017

the algorithm in step 3 of the diagram above

> 30,000
citations
as of August 2026

One 2017 paper, nine pages long. It is the default policy-optimization method at OpenAI, the trainer behind ChatGPT's step 3, and the starting point for the methods that train reasoning models. We are going to derive it.

Semantic Scholar reported 30,354 citations on 21 August 2026; the card rounds. Google Scholar's own count is larger because it indexes more venues.

Where We Left Off

Lecture 20 solved the one-step problem. With a bandit, the score trick turned a gradient we could not compute into an expectation we could sample:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a) \cdot \{r(a) - b\}].$$

Where We Left Off

Lecture 20 solved the one-step problem. With a bandit, the score trick turned a gradient we could not compute into an expectation we could sample:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a) \cdot \{r(a) - b\}].$$

But almost nothing is one step. Even the gridworld of Lecture 20 takes several moves before anything pays, and a Go game takes hundreds.

So the job is to extend this formula from a bandit to a full Markov decision process — and then to keep extending it until it becomes PPO.

Notation We Carry Over: the Objective

An MDP $(\mathcal{S}, \mathcal{A}, P, r, \mu, \gamma)$ and a policy π together generate a trajectory,

$$s_0 \sim \mu, \qquad a_t \sim \pi(\cdot \mid s_t), \qquad s_{t+1} \sim P(\cdot \mid s_t, a_t),$$

and the quantity we are trying to make large is the expected discounted return

$$J(\pi) = \mathbb{E}_{\pi}\Big[\sum_{t \geq 0} \gamma^t \cdot r(s_t, a_t)\Big] = \mathbb{E}_{s_0 \sim \mu}[V^{\pi}(s_0)].$$

Two readings of the same number: average the discounted rewards along a path, or average the value of the states we start in. The second form is the one the lemma will use.

Notation We Carry Over: Values and Advantages

With $G_t = \sum_{k \geq 0} \gamma^k \cdot r_{t+k}$ the return from step t,

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s], \qquad Q^\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s,\, a_t = a],$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \quad : \text{is this action better than my habit here?}$$

Notation We Carry Over: Values and Advantages

With $G_t = \sum_{k \geq 0} \gamma^k \cdot r_{t+k}$ the return from step t,

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_t = s], \qquad Q^\pi(s,a) = \mathbb{E}_\pi[G_t \mid s_t = s,\, a_t = a],$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \quad : \text{is this action better than my habit here?}$$

Two facts we use constantly:

- $V^\pi(s) = \sum_a \pi(a \mid s) \cdot Q^\pi(s,a)$ — the value is the policy's own average of its action values;
- Hence $\sum_a \pi(a \mid s) \cdot A^\pi(s,a) = 0$: the advantage is centred at every state.

The second fact is what makes the whole derivation work. Keep it in view.

Problem Setup

Restrict behaviour to a parametric family $\{\pi_\theta : \theta \in \mathbb{R}^d\}$ and maximize the objective directly:

$$\max_{\theta \in \mathbb{R}^d} J(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}}\Big[\sum_{t \geq 0} \gamma^t \cdot r(s_t, a_t)\Big]$$

Problem Setup

Restrict behaviour to a parametric family $\{\pi_\theta : \theta \in \mathbb{R}^d\}$ and maximize the objective directly:

$$\max_{\theta \in \mathbb{R}^d} J(\theta) = \mathbb{E}_{\tau \sim p_{\pi_\theta}}\Big[\sum_{t \geq 0} \gamma^t \cdot r(s_t, a_t)\Big]$$

- **Model class**: π_θ, a softmax over network logits, exactly as in Lecture 8.
- **Objective**: $J(\theta)$ — but the expectation is over a trajectory law p_{π_θ} that moves with θ.
- **Algorithm**: $\theta \leftarrow \theta + \eta \nabla_\theta J(\theta)$, as always, if the gradient can be computed.

What Makes This Hard

- The environment P is unknown, and usually not differentiable — a simulator, a game, a grader.
- The distribution being averaged over depends on θ, so the gradient does not pass through to the model the way it did for a fixed training set.
- All we can do is run the policy and observe what happens.

What Makes This Hard

- The environment P is unknown, and usually not differentiable — a simulator, a game, a grader.
- The distribution being averaged over depends on θ, so the gradient does not pass through to the model the way it did for a fixed training set.
- All we can do is run the policy and observe what happens.

Lecture 20 solved this for one step. The rest of this lecture is four successive answers to "what is the gradient, and how do I estimate it well?"

Outline

Recall: the Bandit Gradient

With a one-step horizon, Lecture 20 obtained

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a) \cdot r(a)].$$

Recall: the Bandit Gradient

With a one-step horizon, Lecture 20 obtained

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a) \cdot r(a)].$$

Read it as an instruction: raise the log probability of the action you played, in proportion to what it paid.

The environment never appears. That is what made the formula usable, and it is what we want to keep.

The Only Change That Is Needed

In a bandit, "what the action paid" is the single reward $r(a)$. With a horizon, an action is followed by everything that comes after it, so the weight becomes the **return from that step onward**:

$$r(a) \longrightarrow G_t = \sum_{k \geq 0} \gamma^k \cdot r_{t+k}.$$

The Only Change That Is Needed

In a bandit, "what the action paid" is the single reward $r(a)$. With a horizon, an action is followed by everything that comes after it, so the weight becomes the **return from that step onward**:

$$r(a) \longrightarrow G_t = \sum_{k \geq 0} \gamma^k \cdot r_{t+k}.$$

Everything else is unchanged. This gives the **REINFORCE** gradient,

$$\boxed{\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\Big[\sum_{t \geq 0} \gamma^t \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot G_t\Big]}$$

Source: Williams (1992); derived from the trajectory likelihood in Appendix A.1.

Why the Weight Is the Return and Not the Whole Reward

An action at time t cannot change a reward collected before it. Those earlier rewards are independent of a_t, so their contribution to the expectation is zero and they can be dropped:

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot r_{t'}] = 0 \qquad \text{for } t' < t.$$

Why the Weight Is the Return and Not the Whole Reward

An action at time t cannot change a reward collected before it. Those earlier rewards are independent of a_t, so their contribution to the expectation is zero and they can be dropped:

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot r_{t'}] = 0 \qquad \text{for } t' < t.$$

What is left in front of each score is exactly the **reward-to-go** G_t.

A complete, unbiased algorithm: run the policy, compute each G_t, and ascend. It needs no model, no critic, and no new theory.

The Estimator Is Correct and Very Noisy

Write one sampled gradient out and count what goes into it:

$$\hat{g} = \underbrace{\sum_{t \geq 0} \gamma^t}_{\approx 1/(1-\gamma) \text{ terms}} \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \underbrace{\Big(\sum_{k \geq 0} \gamma^k \cdot r_{t+k} \Big)}_{G_t\text{: } \approx 1/(1-\gamma) \text{ rewards}}$$

The Estimator Is Correct and Very Noisy

Write one sampled gradient out and count what goes into it:

$$\hat{g} = \underbrace{\sum_{t \geq 0} \gamma^t}_{\approx 1/(1-\gamma) \text{ terms}} \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \underbrace{\Big(\sum_{k \geq 0} \gamma^k \cdot r_{t+k} \Big)}_{G_t\text{: } \approx 1/(1-\gamma) \text{ rewards}}$$

Both sums run for about one effective horizon, so $\hat{g}$ adds up on the order of $1/(1-\gamma)^2$ reward terms: 10^2 at $\gamma = 0.9$, 10^4 at $\gamma = 0.99$, 10^6 at $\gamma = 0.999$.

The Estimator Is Correct and Very Noisy

Write one sampled gradient out and count what goes into it:

$$\hat{g} = \underbrace{\sum_{t \geq 0} \gamma^t}_{\approx 1/(1-\gamma) \text{ terms}} \cdot \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \underbrace{\Big(\sum_{k \geq 0} \gamma^k \cdot r_{t+k} \Big)}_{G_t\text{: } \approx 1/(1-\gamma) \text{ rewards}}$$

Both sums run for about one effective horizon, so $\hat{g}$ adds up on the order of $1/(1-\gamma)^2$ reward terms: 10^2 at $\gamma = 0.9$, 10^4 at $\gamma = 0.99$, 10^6 at $\gamma = 0.999$.

Every reward is random, so $\hat{g}$ is noisy: correct on average, but the signal about one action is buried in the noise of the whole episode. The lecture builds up to PPO, which trades variance away by accepting bias.

Route to Deriving PPO

1. REINFORCE is unbiased but noisy, so ask instead what the exact improvement from π to π' is. Answer: **performance difference lemma**.
2. Its infinitesimal limit is the **policy gradient theorem**, which gives **actor–critic**.
3. Optimizing the identity itself, with the step held small, gives **PPO**.

Companions to these slides: OpenAI's Spinning Up in Deep RL, and the Hugging Face Deep RL Course unit on PPO.

Outline

What a Descent Direction Actually Is

In optimization, a descent direction comes from a first-order expansion,

$$f(x+\delta) - f(x) = \langle \nabla f(x),\, \delta \rangle + o(\|\delta\|).$$

What a Descent Direction Actually Is

In optimization, a descent direction comes from a first-order expansion,

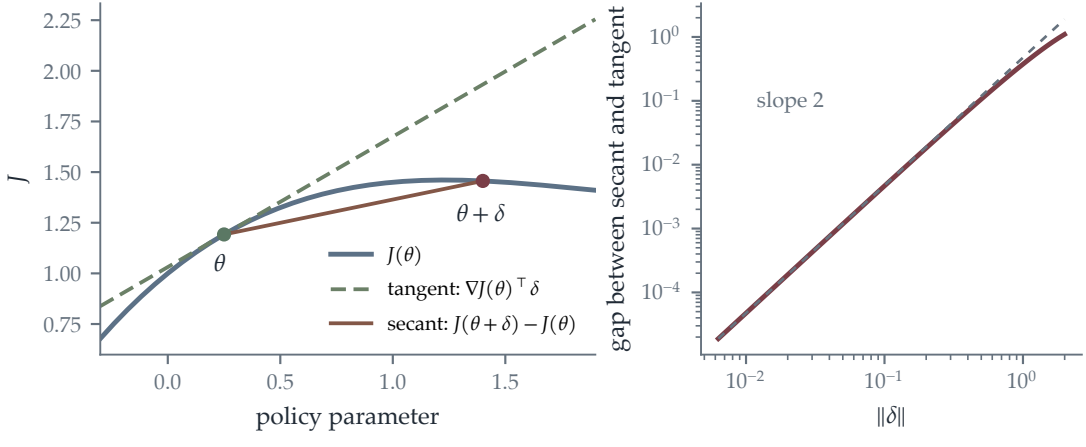
$$f(x+\delta) - f(x) = \langle \nabla f(x),\, \delta \rangle + o(\|\delta\|).$$

Two objects, and they are not the same:

- The finite difference $f(x+\delta) - f(x)$: exact, for any step;
- The gradient $\nabla f(x)$: a local slope, valid as $\delta \to 0$.

We will get the first one exactly for policies, and then take the limit. Doing it in that order is what makes PPO fall out at the end.

The Difference and Its Limit



Left: the secant through two points against the tangent at one. Right: the gap between them shrinks like $\|\delta\|^2$.

The Same Question for Policies

Replace x by a policy and compare a **candidate** π' with a **baseline** π:

$$J(\pi') - J(\pi) = \ ?$$

The Same Question for Policies

Replace x by a policy and compare a **candidate** π' with a **baseline** π:

$$J(\pi') - J(\pi) = \ ?$$

In supervised learning this would be easy: both models are scored on the same held-out set. Here the two policies produce different trajectories, so there is no common sample to compare them on.

Remarkably, there is still an exact answer.

Where a Policy Spends Its Time

Collect the discounted chance of standing in each state:

$$d_\mu^\pi(s) = (1-\gamma) \sum_{t \geq 0} \gamma^t \cdot \mathbb{P}_\pi(s_t = s \mid s_0 \sim \mu).$$

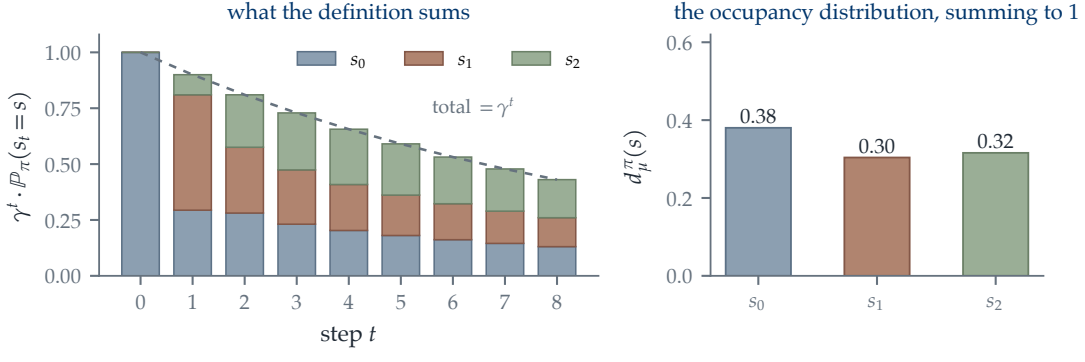
Where a Policy Spends Its Time

Collect the discounted chance of standing in each state:

$$d_\mu^\pi(s) = (1-\gamma) \sum_{t \geq 0} \gamma^t \cdot \mathbb{P}_\pi(s_t = s \mid s_0 \sim \mu).$$

The weights γ^t sum to $1/(1-\gamma)$, so the factor $1-\gamma$ out front makes d_μ^π a probability distribution over states: the **occupancy distribution**, the reinforcement-learning replacement for "the data distribution".

Building the Occupancy Distribution



Left: each step contributes $\gamma^t \cdot \mathbb{P}_\pi(s_t = s)$, so the stack height is γ^t and later steps count for less. Right: adding the columns and normalizing gives one distribution over states — where this policy spends its discounted time.

Exact calculation in the three-state MDP of this lecture; state marginals from powers of P^π, $\gamma = .9$.

Performance Difference Lemma

For any two policies in the same discounted MDP,

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi'}} \Big[\sum_a \{\pi'(a \mid s) - \pi(a \mid s)\} \cdot Q^\pi(s, a) \Big]$$

Performance Difference Lemma

For any two policies in the same discounted MDP,

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma}\,\mathbb{E}_{s\sim d_\mu^{\pi'}}\Big[\sum_a \{\pi'(a\mid s) - \pi(a\mid s)\}\cdot Q^{\pi}(s,a)\Big]$$

Read the bracket as an inner product over actions:

$$\Big[\,\langle Q^{\pi}(s,\cdot),\ \pi'(\cdot\mid s) - \pi(\cdot\mid s)\rangle\,\Big].$$

Everything scored is baseline; everything chosen is candidate.

Source: Kakade and Langford (2002); derived line by line in Appendix B.

This Is the Finite Difference, Exactly

Put the two statements side by side.

$$f(x+\delta) - f(x) \;\approx\; \langle \nabla f(x),\, \delta \rangle$$

$$J(\pi') - J(\pi) \;=\; \frac{1}{1-\gamma}\, \mathbb{E}_{s \sim d^{\pi'}} \Big[\, \langle Q^{\pi}(s,\cdot),\, \pi'(\cdot \mid s) - \pi(\cdot \mid s) \rangle \,\Big]$$

This Is the Finite Difference, Exactly

Put the two statements side by side.

$$f(x+\delta) - f(x) \;\approx\; \langle \nabla f(x),\, \delta \rangle$$

$$J(\pi') - J(\pi) \;=\; \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}}\Big[\, \langle Q^{\pi}(s,\cdot),\, \pi'(\cdot \mid s) - \pi(\cdot \mid s) \rangle \,\Big]$$

calculus	policies
the gradient $\nabla f(x)$	the action values $Q^{\pi}(s,\cdot)$
the step δ	the policy change $\pi'(\cdot \mid s) - \pi(\cdot \mid s)$
$\approx$, error $o(\|\delta\|)$	$=$, no error at all

Reading the Identity

- Q^π is the gradient-like object: it scores each action under the baseline we already have, and it is estimable from baseline data.
- $\pi' - \pi$ is the step: how much probability we move, and where.
- $d^{\pi'}$ is the catch: states are weighted by where the candidate goes, which we cannot sample without running it.

Reading the Identity

- Q^π is the gradient-like object: it scores each action under the baseline we already have, and it is estimable from baseline data.
- $\pi' - \pi$ is the step: how much probability we move, and where.
- $d^{\pi'}$ is the catch: states are weighted by where the candidate goes, which we cannot sample without running it.

The exactness is bought entirely by that third term. Every algorithm in this lecture is a way of coping with it.

The Same Identity with Advantages

derive

Because $\sum_a \pi(a \mid s) \cdot Q^\pi(s,a) = V^\pi(s)$ and $\sum_a \pi'(a \mid s) \cdot V^\pi(s) = V^\pi(s)$, subtracting $V^\pi(s)$ inside the bracket changes nothing:

$$\sum_a \{\pi' - \pi\}(a \mid s) \cdot Q^\pi(s,a) = \sum_a \pi'(a \mid s) \cdot A^\pi(s,a).$$

The Same Identity with Advantages

derive

Because $\sum_a \pi(a \mid s) \cdot Q^\pi(s,a) = V^\pi(s)$ and $\sum_a \pi'(a \mid s) \cdot V^\pi(s) = V^\pi(s)$, subtracting $V^\pi(s)$ inside the bracket changes nothing:

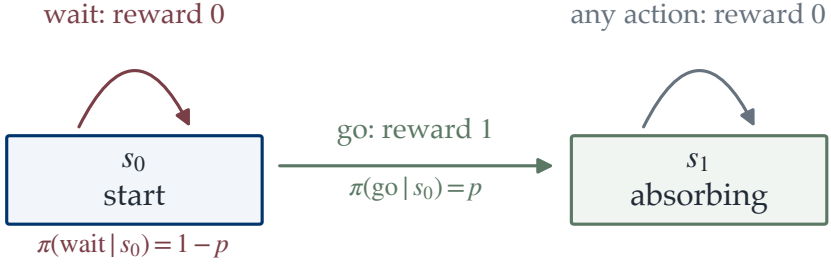
$$\sum_a \{\pi' - \pi\}(a \mid s) \cdot Q^\pi(s,a) = \sum_a \pi'(a \mid s) \cdot A^\pi(s,a).$$

So the lemma also reads

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma}\, \mathbb{E}_{s \sim d_\mu^{\pi'}} \mathbb{E}_{a \sim \pi'(\cdot|s)} [A^\pi(s,a)].$$

Same content, two readings: a difference of policies against Q, or an average of advantages under π'. We use the first to see the geometry and the second to build estimators.

The Two-State Example Again



Baseline π: always wait. Candidate π': go with probability $p = .5$. $\gamma = .9$.

Source: Original course example, continued from Lecture 20.

The Left-Hand Side: Two Returns

Waiting forever collects nothing, so $J(\pi) = 0$. The candidate earns 1 the first time it goes, and reaches s_0 at time t with probability $(1-p)^t$:

$$J(\pi') = \sum_{t \geq 0} \gamma^t \cdot (1-p)^t \cdot p = \frac{p}{1-\gamma(1-p)} = \frac{.5}{1-.45} = \frac{10}{11}.$$

The Left-Hand Side: Two Returns

Waiting forever collects nothing, so $J(\pi) = 0$. The candidate earns 1 the first time it goes, and reaches s_0 at time t with probability $(1-p)^t$:

$$J(\pi') = \sum_{t \geq 0} \gamma^t \cdot (1-p)^t \cdot p = \frac{p}{1 - \gamma(1-p)} = \frac{.5}{1 - .45} = \frac{10}{11}.$$

So the quantity the lemma must reproduce is

$$J(\pi') - J(\pi) = \frac{10}{11} \approx .909.$$

The Right-Hand Side: Advantages of the Baseline

Everything on the right is computed from π alone. It never moves, so $V^\pi \equiv 0$ and Q^π is just the immediate reward:

$$A^\pi(s_0, \text{go}) = 1 - 0 = 1, \qquad A^\pi(s_0, \text{wait}) = 0, \qquad A^\pi(s_1, \cdot) = 0.$$

The Right-Hand Side: Advantages of the Baseline

Everything on the right is computed from π alone. It never moves, so $V^\pi \equiv 0$ and Q^π is just the immediate reward:

$$A^\pi(s_0, \text{go}) = 1 - 0 = 1, \qquad A^\pi(s_0, \text{wait}) = 0, \qquad A^\pi(s_1, \cdot) = 0.$$

Averaging these under the candidate's own action probabilities,

$$\mathbb{E}_{a \sim \pi'(\cdot|s_0)}[A^\pi(s_0, a)] = p \cdot 1 + (1 - p) \cdot 0 = .5.$$

The Right-Hand Side: Where the Candidate Goes

The candidate is at s_0 at time t with probability $(1-p)^t$, so its **occupancy distribution** is

$$d^{\pi'}(s_0) = (1-\gamma)\sum_{t\geq 0}[\gamma(1-p)]^t = \frac{.1}{.55} = \frac{2}{11}, \qquad d^{\pi'}(s_1) = \frac{9}{11}.$$

The Right-Hand Side: Where the Candidate Goes

The candidate is at s_0 at time t with probability $(1-p)^t$, so its **occupancy distribution** is

$$d^{\pi'}(s_0) = (1-\gamma)\sum_{t\geq 0}[\gamma(1-p)]^t = \frac{.1}{.55} = \frac{2}{11}, \qquad d^{\pi'}(s_1) = \frac{9}{11}.$$

Assembling the two pieces:

$$\frac{1}{1-\gamma}\Big[\tfrac{2}{11}\cdot .5 + \tfrac{9}{11}\cdot 0\Big] = 10\cdot\frac{1}{11} = \frac{10}{11} \;=\; J(\pi') - J(\pi). \quad \checkmark$$

The advantage alone would have said "+1". Only 2/11 of the candidate's discounted time is spent where that advantage is available — and the identity is exact once that is counted.

Outline

Take the Step to Zero

The lemma is the finite difference. The gradient is what it becomes when the step shrinks. So put $\pi = \pi_\theta$, $\pi' = \pi_{\theta+\delta}$ in

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi'}} \Big[\langle Q^{\pi_\theta}(s, \cdot),\ \pi'(\cdot \mid s) - \pi(\cdot \mid s) \rangle \Big]$$

and let $\delta \to 0$.

Take the Step to Zero

The lemma is the finite difference. The gradient is what it becomes when the step shrinks. So put $\pi = \pi_\theta$, $\pi' = \pi_{\theta+\delta}$ in

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi'}} \Big[\langle Q^{\pi_\theta}(s, \cdot),\ \pi'(\cdot \mid s) - \pi(\cdot \mid s) \rangle \Big]$$

and let $\delta \to 0$.

Two things depend on δ: the policy difference inside, and the occupancy distribution outside. We analyze them separately.

Swapping the Occupancy Distribution Is Higher Order

The obstacle was the outside weighting: states come from $d^{\pi'}$, which we cannot sample. As $\delta \to 0$ the two occupancy distributions coincide — but what we need is the rate, not just the limit. The error made by swapping them is

$$\frac{1}{1-\gamma} \sum_s \underbrace{\{d^{\pi'}(s) - d^{\pi_\theta}(s)\}}_{O(\|\delta\|)} \cdot \underbrace{\langle Q^{\pi_\theta}(s,\cdot),\ \pi'(\cdot \mid s) - \pi_\theta(\cdot \mid s)\rangle}_{O(\|\delta\|)} \ = \ O(\|\delta\|^2).$$

Swapping the Occupancy Distribution Is Higher Order

The obstacle was the outside weighting: states come from $d^{\pi'}$, which we cannot sample. As $\delta \to 0$ the two occupancy distributions coincide — but what we need is the rate, not just the limit. The error made by swapping them is

$$\frac{1}{1-\gamma} \sum_s \underbrace{\{d^{\pi'}(s) - d^{\pi_\theta}(s)\}}_{O(\|\delta\|)} \cdot \underbrace{\langle Q^{\pi_\theta}(s, \cdot),\ \pi'(\cdot \mid s) - \pi_\theta(\cdot \mid s) \rangle}_{O(\|\delta\|)} \ = \ O(\|\delta\|^2).$$

A higher-order term: it cannot affect a first-order expansion.

States may be weighted by the baseline's own occupancy distribution — the one we can sample. This is the single fact that makes policy gradients practical.

Source: Both rates are checked in Appendix C.1.

The Policy Difference Becomes a Score

derive

That leaves the inside. Expand it to first order, then use the score identity $\nabla_\theta \pi_\theta = \pi_\theta \cdot \nabla_\theta \log \pi_\theta$ from Lecture 20:

$$\begin{aligned} \pi_{\theta+\delta}(a \mid s) - \pi_\theta(a \mid s) &= \nabla_\theta \pi_\theta(a \mid s)^\top \delta + o(\|\delta\|) \\ &= \pi_\theta(a \mid s) \cdot \nabla_\theta \log \pi_\theta(a \mid s)^\top \delta + o(\|\delta\|). \end{aligned}$$

The same one-line trick as the bandit case, doing the same job: turning a sum over all actions into an expectation over the action actually taken.

Policy Gradient Theorem

Substituting both facts and matching against $J(\theta + \delta) - J(\theta) = \langle \nabla_\theta J,\, \delta \rangle + o(\|\delta\|)$:

$$\boxed{\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\, \mathbb{E}_{s \sim d^{\pi_\theta},\, a \sim \pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s, a) \right]}$$

Policy Gradient Theorem

Substituting both facts and matching against $J(\theta + \delta) - J(\theta) = \langle \nabla_\theta J,\, \delta \rangle + o(\|\delta\|)$:

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\, \mathbb{E}_{s \sim d^{\pi_\theta},\, a \sim \pi_\theta(\cdot|s)} \left[\nabla_\theta \log \pi_\theta(a \mid s) \cdot Q^{\pi_\theta}(s,a) \right]$$

P does not appear explicitly. It is hidden inside d^{π_θ} and Q^{π_θ} — but only as something we sample. Nothing in the formula asks us to know P.

Source: Sutton et al. (2000); full derivation in Appendix A.2.

Any Baseline May Be Subtracted

derive

Let $b : \mathcal{S} \to \mathbb{R}$ be any function of the state — it may not depend on the action. Then, at each fixed s,

$$\mathbb{E}_{a \sim \pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s) \cdot b(s)] = b(s) \cdot \sum_a \nabla_\theta \pi_\theta(a \mid s) = b(s) \cdot \nabla_\theta 1 = 0.$$

Any Baseline May Be Subtracted

derive

Let $b : \mathcal{S} \to \mathbb{R}$ be any function of the state — it may not depend on the action. Then, at each fixed s,

$$\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s) \cdot b(s)] = b(s) \cdot \sum_a \nabla_\theta \pi_\theta(a \mid s) = b(s) \cdot \nabla_\theta 1 = 0.$$

Subtracting it therefore changes nothing in expectation:

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma}\, \mathbb{E}_{s\sim d^{\pi_\theta},\, a\sim\pi_\theta(\cdot|s)}[\nabla_\theta \log \pi_\theta(a \mid s) \cdot \{Q^{\pi_\theta}(s,a) - b(s)\}].$$

One unbiased gradient, a whole family of estimators. They differ only in variance — which is the entire reason to introduce b. We have seen this in the last lecture.

The Default Baseline Is the Value Function

Take $b(s) = V^{\pi_\theta}(s)$, the average of $Q^{\pi_\theta}(s, \cdot)$ under π_θ. Then $Q^{\pi_\theta} - b$ is exactly the advantage:

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta},\, a \sim \pi_\theta(\cdot|s)} [\nabla_\theta \log \pi_\theta(a \mid s) \cdot A^{\pi_\theta}(s, a)]$$

The Default Baseline Is the Value Function

Take $b(s) = V^{\pi_\theta}(s)$, the average of $Q^{\pi_\theta}(s, \cdot)$ under π_θ. Then $Q^{\pi_\theta} - b$ is exactly the advantage:

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi_\theta},\, a \sim \pi_\theta(\cdot|s)} [\nabla_\theta \log \pi_\theta(a \mid s) \cdot A^{\pi_\theta}(s, a)]$$

Why this choice: A^{π_θ} is centred at every state. Actions better than the current average get a positive weight, worse ones a negative weight.

Without it, if all rewards are positive then every log-probability is pushed up, and the useful signal is the small difference between large numbers.

What Changed Since REINFORCE

REINFORCE

weight = G_t, a sampled return: a sum of about $\frac{1}{1-\gamma}$ random rewards

Policy gradient theorem

weight = $Q^{\pi_\theta}(s,a)$ or $A^{\pi_\theta}(s,a)$, the mean of that return

What Changed Since REINFORCE

REINFORCE

weight $= G_t$, a sampled return: a sum of about $\frac{1}{1-\gamma}$ random rewards

Policy gradient theorem

weight $= Q^{\pi_\theta}(s,a)$ or $A^{\pi_\theta}(s,a)$, the mean of that return

Same expectation. The second replaces a noisy sum by the number it averages to — which removes the variance REINFORCE was carrying.

At a price: Q^{π_θ} and A^{π_θ} are not given to us. Something has to estimate them.

Consistency Check: the Bandit Case

Set $\gamma = 0$. The occupancy distribution collapses to the initial one, $d^{\pi_\theta} = \mu$, and $Q^{\pi_\theta}(s,a) = r(s,a)$. The theorem reads

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\mu,\, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) \cdot r(s,a)],$$

which is exactly where Lecture 20 stopped.

Consistency Check: the Bandit Case

Set $\gamma = 0$. The occupancy distribution collapses to the initial one, $d^{\pi_\theta} = \mu$, and $Q^{\pi_\theta}(s,a) = r(s,a)$. The theorem reads

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\mu,\, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a \mid s) \cdot r(s,a)],$$

which is exactly where Lecture 20 stopped.

The general theorem contains the bandit result as a special case. The two new ingredients are the occupancy distribution d^{π_θ} and the long-run value in place of an immediate reward.

Outline

The Algorithm the Theorem Suggests

The policy gradient needs A^{π_θ}, which we do not have. But the advantage can be written as a one-step residual,

$$A^{\pi}(s_t, a_t) = \mathbb{E}\left[\, r_t + \gamma \cdot V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \mid s_t, a_t \,\right],$$

The Algorithm the Theorem Suggests

The policy gradient needs A^{π_θ}, which we do not have. But the advantage can be written as a one-step residual,

$$A^{\pi}(s_t, a_t) = \mathbb{E}\big[\, r_t + \gamma \cdot V^{\pi}(s_{t+1}) - V^{\pi}(s_t) \mid s_t, a_t \,\big],$$

so a single observed transition (s_t, a_t, r_t, s_{t+1}) estimates it — as soon as we have V^{π}. Estimate V^{π} from the data the policy is already generating, and weight the score by the residual.

actor π_θ	moves along $\nabla_\theta \log \pi_\theta \cdot \widehat{A}$
critic V_ϕ	fitted by least squares to the returns it sees

The Value Function Is a Regression Target

The Bellman equation says

$$V^{\pi}(s) = \mathbb{E}\left[\, r_t + \gamma \cdot V^{\pi}(s_{t+1}) \mid s_t = s \,\right].$$

The Value Function Is a Regression Target

The Bellman equation says

$$V^{\pi}(s) = \mathbb{E}\left[\, r_t + \gamma \cdot V^{\pi}(s_{t+1}) \mid s_t = s \,\right].$$

A conditional expectation is exactly what least squares estimates. Fit a network V_{ϕ} by

$$\min_{\phi} \sum_{t} \left(V_{\phi}(s_t) - y_t\right)^2, \qquad y_t = r_t + \gamma \cdot V^{\pi}(s_{t+1}).$$

Except that the target y_t contains the very function we are trying to estimate.

Use the Previous Estimate as the Target

Substitute a **frozen copy** ϕ^- of the current parameters:

$$\min_{\phi} \sum_{t} \Big(V_{\phi}(s_t) - \underbrace{\{r_t + \gamma \cdot V_{\phi}^{-}(s_{t+1})\}}_{\text{target, treated as a constant}} \Big)^2$$

Use the Previous Estimate as the Target

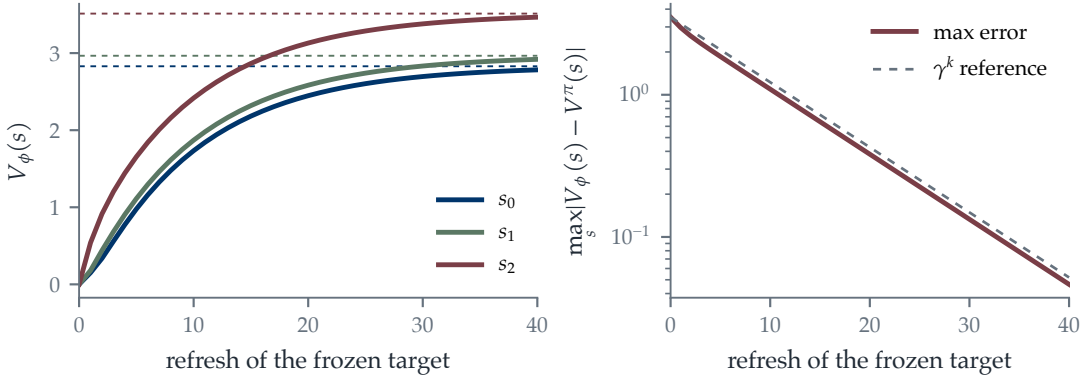
Substitute a **frozen copy** ϕ^- of the current parameters:

$$\min_{\phi} \sum_{t} \Big(V_{\phi}(s_t) - \underbrace{\{r_t + \gamma \cdot V_{\phi}^{-}(s_{t+1})\}}_{\text{target, treated as a constant}} \Big)^2$$

Now it is an ordinary regression, and it is run as one: compute every target once, then take many minibatch steps on ϕ against those same numbers. Only when that fit has converged do we set $\phi^- \leftarrow \phi$ and recompute the targets.

Unfrozen, the regression would chase its own output. The frozen copy is a target network, and the two timescales are the trick: ϕ moves every step, ϕ^- only between rounds.

The Critic Converges at the Discount Rate



Refitting against a frozen copy, in the three-state MDP. Left: the estimates climb to the true V^π (dashed). Right: the largest error falls exactly along γ^k — the same contraction that made $(I - \gamma P^\pi)$ invertible in Lecture 20, now doing the work numerically.

$\gamma = .9$, tabular least squares so each refresh is one exact backup.

What This Costs

Complete return

wait until the episode ends and use $\sum_k \gamma^k \cdot r_{t+k}$: unbiased, but high variance and only available at the end

Bootstrapped target

use $r_t + \gamma \cdot V_\phi^-(s_{t+1})$: available immediately, far lower variance, biased while V_ϕ is wrong

What This Costs

Complete return

wait until the episode ends and use $\sum_k \gamma^k \cdot r_{t+k}$: unbiased, but high variance and only available at the end

Bootstrapped target

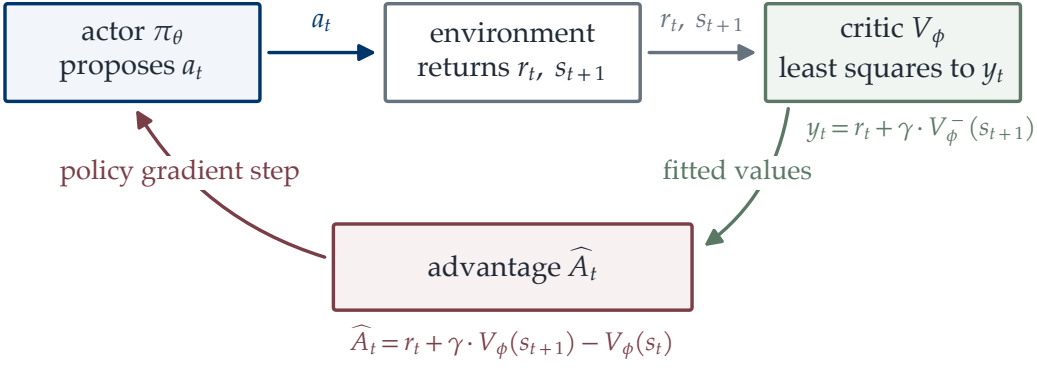
use $r_t + \gamma \cdot V_{\phi}^{-}(s_{t+1})$: available immediately, far lower variance, biased while V_{ϕ} is wrong

The same bias–variance trade-off as everywhere else in the course, now applied to the target rather than to the model.

A smooth interpolation between the two extremes exists — **generalized advantage estimation** (GAE) — and is what PPO implementations use. We do not develop it here.

Source: Schulman et al. (2016), arXiv:1506.02438.

The Actor–Critic Loop



The actor supplies the data, the critic scores it, and the advantage sends a low-variance weight back — without waiting for an episode to end.

The actor is optimized by the policy gradient; the critic is fitted by least squares onto the frozen target ϕ^-.

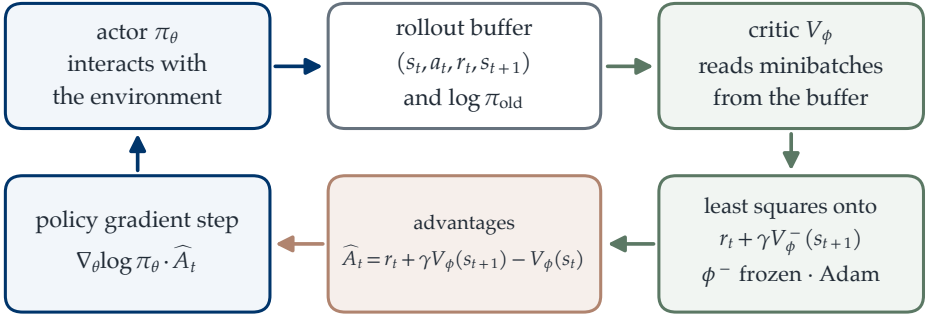
The Actor–Critic Algorithm

Repeat:

1. Run π_θ for a batch of steps, recording (s_t, a_t, r_t, s_{t+1});
2. Compute advantages $\widehat{A}_t = r_t + \gamma \cdot V_\phi(s_{t+1}) - V_\phi(s_t)$;
3. Actor: $\theta \leftarrow \theta + \eta_\theta \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \widehat{A}_t$;
4. Critic: hold ϕ^- fixed and run K minibatch steps on $\phi \leftarrow \arg\min_\phi \sum_t (V_\phi(s_t) - \{r_t + \gamma \cdot V_{\phi}^{-}(s_{t+1})\})^2$;
5. Refresh the target network, $\phi^- \leftarrow \phi$.

Two networks, two objectives, one stream of data. In a language model both usually share a trunk, with the critic a scalar head on top.

What This Looks Like in Practice



θ moves, so the buffer goes stale:
discard it and refill

The policy fills a **rollout buffer**; the critic reads it as an ordinary supervised data set, with the target held fixed for the whole fit.

The Critic Trades Bias for Variance

REINFORCE weights the score by a sampled return, which is unbiased, $\mathbb{E}[G_t \mid s_t = s, a_t = a] = Q^\pi(s,a)$, but G_t adds up about $1/(1-\gamma)$ random rewards. The residual adds three terms:

$$\widehat{A}_t = r_t + \gamma \cdot V_\phi(s_{t+1}) - V_\phi(s_t), \qquad \text{one reward, two lookups.}$$

The Critic Trades Bias for Variance

REINFORCE weights the score by a sampled return, which is unbiased, $\mathbb{E}[G_t \mid s_t = s, a_t = a] = Q^\pi(s,a)$, but G_t adds up about $1/(1-\gamma)$ random rewards. The residual adds three terms:

$$\widehat{A}_t = r_t + \gamma \cdot V_\phi(s_{t+1}) - V_\phi(s_t), \qquad \text{one reward, two lookups.}$$

The price is a bias, $\widehat{A}_t - A^\pi(s_t, a_t)$: exactly the critic's fit error.

G_t carries the randomness of the whole episode; the residual carries one step of it. A bias we can shrink by fitting better, for a variance we cannot otherwise remove.

The Remaining Weakness

The actor step is a single small move along a gradient estimated from one batch of rollouts. After the move, θ has changed, so the batch is no longer on-policy and must be thrown away.

- Rollouts are the expensive part — a simulator, a game, or a language-model generation.
- A large step reuses the batch better, but leaves the regime where the gradient is valid.

The Remaining Weakness

The actor step is a single small move along a gradient estimated from one batch of rollouts. After the move, θ has changed, so the batch is no longer on-policy and must be thrown away.

- Rollouts are the expensive part — a simulator, a game, or a language-model generation.
- A large step reuses the batch better, but leaves the regime where the gradient is valid.

We want to take several steps on one batch, and to know when to stop. PPO fixes these issues.

Outline

Start from the Exact Identity

We want to maximize over π', with π the current policy:

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s\sim d^{\pi'}} \mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)].$$

Start from the Exact Identity

We want to maximize over π', with π the current policy:

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d^{\pi'}} \mathbb{E}_{a \sim \pi'(\cdot|s)} [A^{\pi}(s,a)].$$

Two obstacles, and one approximation each.

- $s \sim d^{\pi'}$: we would have to run the candidate to get these states.
- $a \sim \pi'$: our logged actions were drawn from the baseline, not the candidate.
- We can only sample trajectories $\{(s_t, a_t)\}_{t \geq 0}$ from the current policy $\pi = \pi_{\text{old}}$.

First Approximation: Freeze the Occupancy Distribution

Replace $d^{\pi'}$ by d^{π}, giving the **surrogate objective**

$$L(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)].$$

First Approximation: Freeze the Occupancy Distribution

Replace $d^{\pi'}$ by d^{π}, giving the **surrogate objective**

$$L(\pi') = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi}}\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)].$$

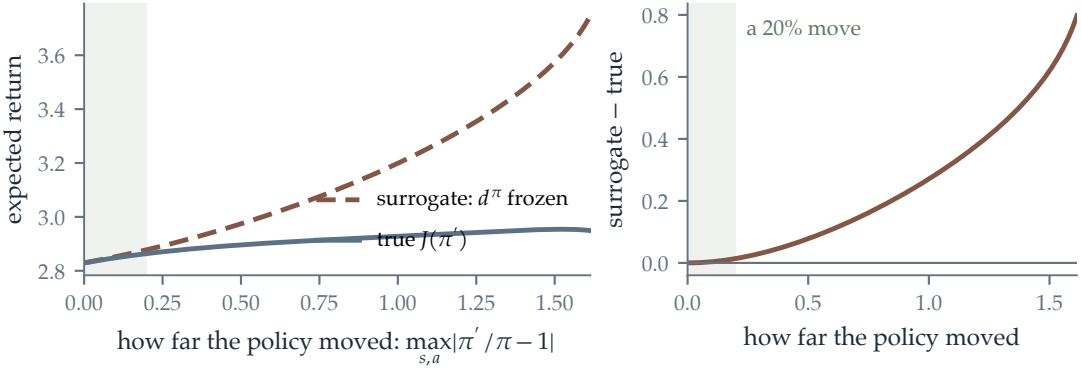
We already know the cost of this swap: the discarded term is $O(\|\pi' - \pi\|^2)$. So

$$L(\pi') \approx J(\pi') - J(\pi) \qquad \text{while the two policies stay close.}$$

Note that we can sample from d^{π} — drawing a trajectories from π and record states.

Approximation is close only if π' is close to π. Our updates will enforce this.

The Surrogate Is Trustworthy Only Near the Baseline



L is an approximation, good only while the policy stays close. Within a 20% move it tracks the true return to .013; far outside, it promises a gain of .80 that does not exist.

Exactly solvable three-state MDP; $\gamma = .9$, values and occupancy distributions solved exactly.

Second Approximation: Reweight the Actions

derive

Inside a fixed state, change the sampling distribution by the standard reweighting identity:

$$\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] = \sum_{a} \pi(a \mid s) \cdot \frac{\pi'(a \mid s)}{\pi(a \mid s)} \cdot A^{\pi}(s,a) = \mathbb{E}_{a\sim\pi}\Big[\frac{\pi'(a \mid s)}{\pi(a \mid s)} \cdot A^{\pi}(s,a)\Big].$$

Second Approximation: Reweight the Actions

derive

Inside a fixed state, change the sampling distribution by the standard reweighting identity:

$$\mathbb{E}_{a\sim\pi'(\cdot|s)}[A^{\pi}(s,a)] = \sum_{a} \pi(a \mid s) \cdot \frac{\pi'(a \mid s)}{\pi(a \mid s)} \cdot A^{\pi}(s,a) = \mathbb{E}_{a\sim\pi}\Big[\frac{\pi'(a \mid s)}{\pi(a \mid s)} \cdot A^{\pi}(s,a)\Big].$$

This step is exact — it is valid wherever π has support over action space $\mathcal{A}$.

Note this importance sampling is not a product of ratios over the whole trajectory. Freezing the occupancy distribution already absorbed the time dimension, so a single-step ratio suffices.

The Surrogate We Can Actually Compute

Write π_{old} for the policy that collected the batch and

$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)}.$$

Then, up to the constant $1/(1-\gamma)$, we can write the objective as

$$\boxed{L(\theta) = \widehat{\mathbb{E}}_t[\rho_t(\theta) \cdot \widehat{A}_t]}$$

with $\widehat{A}_t$ from the critic and $\widehat{\mathbb{E}}_t$ the average over the logged transitions from π_{old}.

Every symbol is a number we can compute. And because θ appears only through ρ_t, we may take many gradient steps on one batch instead of one.

The Rollout Data

t	state	action	$\pi_{\text{old}}(a_t \mid s_t)$	r_t	$\widehat{A}_t$
0	s_0	a_1	.38	.40	+.73
1	s_2	a_1	.57	.20	−.15
2	s_2	a_1	.57	.20	−.64
3	s_1	a_0	.40	.30	+.50
4	s_2	a_0	.43	1.00	+.65

One row is one interaction, with the probability π_{old} assigned to the action it took and the advantage the critic estimated. These are exactly the two numbers the ratio ρ_t and the weight $\widehat{A}_t$ need.

Source: First five steps of a rollout in the three-state MDP; original course simulation, $\gamma = .9$, seed 26521.

Sanity Check: One Step Recovers the Policy Gradient

At $\theta = \theta_{\text{old}}$ every ratio is 1, and

$$\nabla_\theta \rho_t\big|_{\theta_{\text{old}}} = \frac{\nabla_\theta \pi_\theta(a_t \mid s_t)}{\pi_{\text{old}}(a_t \mid s_t)} = \nabla_\theta \log \pi_\theta(a_t \mid s_t).$$

So $\nabla_\theta L(\theta_{\text{old}}) = \widehat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \widehat{A}_t]$ — the policy gradient itself.

Nothing Yet Stops the Step from Being Large

Maximizing $\widehat{\mathbb{E}}_t[\rho_t \cdot \widehat{A}_t]$ without a constraint drives $\rho_t \to \infty$ wherever $\widehat{A}_t > 0$: the objective is linear in the ratio and unbounded above.

Nothing Yet Stops the Step from Being Large

Maximizing $\widehat{\mathbb{E}}_t[\rho_t \cdot \widehat{A}_t]$ without a constraint drives $\rho_t \to \infty$ wherever $\widehat{A}_t > 0$: the objective is linear in the ratio and unbounded above.

But a large ρ_t is exactly the regime where the frozen occupancy distribution is wrong, and where $\widehat{A}_t$ — fitted on baseline data — is least trustworthy.

The optimizer will happily exploit the approximation it was handed. Any fix must make large ratios unrewarding.

A Short History: From Trust Regions to Clipping

- **2002, Kakade and Langford**: the exact identity comparing two policies, and the observation that a small step yields reliable policy improvement.
- **2015, TRPO**: enforce smallness as a constraint, $\mathrm{KL}(\pi_\theta \| \pi_{\text{old}}) \leq \varepsilon$. It works, and it needs a constrained solver.
- **2017, PPO**: get the same effect by changing the objective so a large step stops paying. One extra line of code.

A Short History: From Trust Regions to Clipping

- **2002, Kakade and Langford**: the exact identity comparing two policies, and the observation that a small step yields reliable policy improvement.
- **2015, TRPO**: enforce smallness as a constraint, $\mathrm{KL}(\pi_\theta \| \pi_{\text{old}}) \leq \varepsilon$. It works, and it needs a constrained solver.
- **2017, PPO**: get the same effect by changing the objective so a large step stops paying. One extra line of code.

Fifteen years separate the identity from the algorithm everyone uses. We will cover that distance in one lecture.

Two Ways to Keep the Step Small

Trust region

maximize $L(\theta)$ subject to $\mathbb{E}_s \, \mathrm{KL}(\pi_\theta \| \pi_{\text{old}}) \le \varepsilon$; principled, but needs a constrained solver

Clipping

change the objective so that moving past $\rho_t \in [1-\epsilon, 1+\epsilon]$ buys nothing; plain SGD, one extra line of code

The second is PPO. It is an empirical device, not a theorem — and it is what almost every system in practice uses.

Source: Schulman et al. (2015) for the trust region; Schulman et al. (2017) for the clipped form.

The Clipped Objective

$$L^{\text{CLIP}}(\theta) = \widehat{\mathbb{E}}_t\Big[\min(\rho_t(\theta)\cdot\widehat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\cdot\widehat{A}_t)\Big]$$

The Clipped Objective

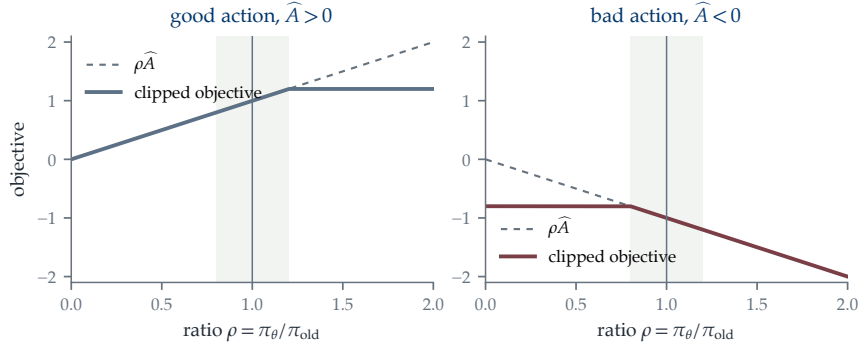
$$L^{\text{CLIP}}(\theta) = \widehat{\mathbb{E}}_t\Big[\min(\rho_t(\theta)\cdot\widehat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\cdot\widehat{A}_t)\Big]$$

Read the two cases separately.

- $\widehat{A}_t > 0$: the term is $\widehat{A}_t \cdot \min(\rho_t, 1+\epsilon)$ — raising the probability helps, but only up to $1+\epsilon$.
- $\widehat{A}_t < 0$: the term is $\widehat{A}_t \cdot \max(\rho_t, 1-\epsilon)$ — lowering the probability helps only down to $1-\epsilon$.

Typically $\epsilon = .2$. The outer min makes the objective a pessimistic version of the surrogate: it never rewards a move more than the unclipped term would.

Where the Gradient Switches Off



One transition's contribution as its ratio moves. Inside the shaded band the objective is the plain surrogate. Outside it, on the side that would keep improving the objective, the curve is flat: that sample contributes zero gradient and stops pushing. On the side that would make things worse, the penalty is left intact.

With $\epsilon = .2$ and $|\widehat{A}| = 1$.

Proximal Policy Optimization

Repeat:

1. Collect a batch of rollouts with the current policy; set $\pi_{\text{old}} \leftarrow \pi_\theta$ and record $\log \pi_{\text{old}}(a_t \mid s_t)$;
2. Compute $\widehat{A}_t$ from the critic, and value targets $r_t + \gamma \cdot V_\phi^-(s_{t+1})$;
3. For a few epochs, take minibatch SGD steps on
$$L^{\text{CLIP}}(\theta) - c \cdot \widehat{\mathbb{E}}_t[(V_\phi(s_t) - y_t)^2];$$
4. Discard the batch and return to step 1.

Steps 1 and 4 are the expensive ones. Clipping is what makes it safe to run step 3 several times per batch.

Outline

Summary: One Identity, Two Readings

$$J(\pi') - J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi'}, a\sim\pi'}[A^{\pi}(s,a)], \qquad \nabla_\theta J = \frac{1}{1-\gamma}\mathbb{E}_{d^{\pi_\theta},\pi_\theta}[\nabla_\theta \log \pi_\theta \cdot A^{\pi_\theta}].$$

1. The performance difference lemma compares two policies exactly, using baseline advantages under the candidate's occupancy distribution.
2. Because the advantage is centred, the change in occupancy distribution is second order, so the gradient uses the baseline's own occupancy distribution.
3. That limit is the policy gradient theorem, and it contains last lecture's bandit formula as the case $\gamma = 0$.

Summary: From the Identity to the Algorithms

$$\widehat{A}_t = r_t + \gamma \cdot V_\phi(s_{t+1}) - V_\phi(s_t), \qquad L^{\text{CLIP}} = \widehat{\mathbb{E}}_t[\min(\rho_t \cdot \widehat{A}_t, \text{clip}(\rho_t, 1 \pm \epsilon) \cdot \widehat{A}_t)].$$

1. The critic is fitted by least squares onto a target built from a frozen copy of itself; the actor follows the gradient with advantage weights.
2. Freezing the occupancy distribution and reweighting the actions turns the exact difference into an objective optimizable for several epochs on one batch.
3. Clipping removes the reward for large ratios, which keeps that objective near the quantity it approximates.

Next Lecture

A policy is a distribution over actions given a state. So is a language model, with the text so far in place of the state.

Lecture 22 — Language Models and Transformers asks where such a model comes from: what training on raw text optimizes, and what architecture makes it possible.

Reading for this lecture: Schulman et al. (2017), *Proximal Policy Optimization Algorithms*.

Appendix

Three groups of notes skipped in the lecture hour.

A. **Policy gradients.** REINFORCE from the trajectory likelihood; why the rewards collected before an action may be dropped; the lemma's infinitesimal limit written out in full.

B. **The performance difference lemma.** The advantage as a one-step residual, the telescoping sum that proves the identity, the time sum rewritten as an occupancy distribution, and the Q form.

C. **The approximations behind PPO.** Why freezing the occupancy distribution costs only $O(\|\delta\|^2)$; the trajectory proof of the policy gradient; why clipping switches the gradient off.

A.1 REINFORCE from the Trajectory Likelihood

derive

Write $J(\theta) = \int G(\tau)\, p_\theta(\tau)\, d\tau$ with $p_\theta(\tau) = \mu(s_0) \prod_t \pi_\theta(a_t \mid s_t) \cdot P(s_{t+1} \mid s_t, a_t)$. Only the policy factors carry θ, so

$$\nabla_\theta \log p_\theta(\tau) = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t),$$

and $\nabla p = p \nabla \log p$ gives

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta}\Big[G(\tau) \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) \Big].$$

The transition law P has already vanished: it does not depend on θ.

A.2 Dropping the Rewards That Came First

derive

In the double sum $G(\tau)\sum_t \nabla \log \pi_\theta(a_t \mid s_t)$, pair each score at time t with each reward at time t'. For $t' < t$ the reward is determined before a_t is drawn, so conditioning on s_t,

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot r_{t'}] = \mathbb{E}\Big[r_{t'} \underbrace{\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \mid s_t]}_{=0}\Big] = 0,$$

because a score has mean zero under its own distribution.

Only the terms with $t' \geq t$ survive, and they sum to $\gamma^t \cdot G_t$. That leaves the reward-to-go form quoted in the lecture.

A.3 From the Lemma to the Policy Gradient

derive

Start from the Q form of the lemma with $\pi' = \pi_{\theta+\delta}$, freeze the occupancy distribution (C.1), and expand:

$$\begin{aligned} J(\theta+\delta) - J(\theta) &= \tfrac{1}{1-\gamma} \textstyle\sum_s d^{\pi_\theta}(s) \sum_a \{\pi_{\theta+\delta} - \pi_\theta\}(a \mid s) \cdot Q^{\pi_\theta}(s,a) + o(\|\delta\|) \\ &= \tfrac{1}{1-\gamma} \textstyle\sum_s d^{\pi_\theta}(s) \sum_a \nabla_\theta \pi_\theta(a \mid s)^\top \delta \cdot Q^{\pi_\theta}(s,a) + o(\|\delta\|) \\ &= \Big\{ \tfrac{1}{1-\gamma} \textstyle\sum_s d^{\pi_\theta}(s) \sum_a \pi_\theta(a \mid s) \cdot \nabla_\theta \log \pi_\theta(a \mid s) Q^{\pi_\theta}(s,a) \Big\}^\top \delta + o(\|\delta\|). \end{aligned}$$

Matching against $\langle \nabla_\theta J,\, \delta \rangle + o(\|\delta\|)$ identifies the brace as $\nabla_\theta J(\theta)$.

B.1 The Advantage as a One-Step Residual

derive

From the Bellman form of Q^π,

$$\begin{aligned} A^\pi(s_t, a_t) &= Q^\pi(s_t, a_t) - V^\pi(s_t) \\ &= \mathbb{E}[r_t + \gamma \cdot V^\pi(s_{t+1}) \mid s_t, a_t] - V^\pi(s_t). \end{aligned}$$

Taking expectations along trajectories generated by the candidate π',

$$\mathbb{E}_{\pi'}[A^\pi(s_t, a_t)] = \mathbb{E}_{\pi'}[r_t + \gamma \cdot V^\pi(s_{t+1}) - V^\pi(s_t)].$$

The baseline value function is evaluated at states the candidate visits.

B.2 Telescoping the Value Terms

derive

Multiply by γ^t and sum over time:

$$\begin{aligned}\sum_{t\geq 0}\gamma^t\cdot\mathbb{E}_{\pi'}[A^\pi(s_t,a_t)] &= J(\pi')+\sum_{t\geq 0}\gamma^{t+1}\cdot\mathbb{E}_{\pi'}V^\pi(s_{t+1})-\sum_{t\geq 0}\gamma^t\cdot\mathbb{E}_{\pi'}V^\pi(s_t)\\ &= J(\pi')-\mathbb{E}_{s_0\sim\mu}V^\pi(s_0)\\ &= J(\pi')-J(\pi).\end{aligned}$$

The two sums are the same series shifted by one index, so everything cancels except the $t=0$ term — which is exactly $J(\pi)$.

B.3 From a Time Sum to an Occupancy Distribution

derive

Regroup the same left-hand side by state and action instead of by time:

$$\begin{aligned}\sum_{t\ge 0}\gamma^t\cdot\mathbb{E}_{\pi'}[A^{\pi}(s_t,a_t)] &= \sum_{t\ge 0}\gamma^t\cdot\sum_{s}\mathbb{P}_{\pi'}(s_t=s)\sum_{a}\pi'(a\mid s)\cdot A^{\pi}(s,a)\\ &= \sum_{s}\Big\{\sum_{t\ge 0}\gamma^t\cdot\mathbb{P}_{\pi'}(s_t=s)\Big\}\sum_{a}\pi'(a\mid s)\cdot A^{\pi}(s,a)\\ &= \frac{1}{1-\gamma}\sum_{s}d_{\mu}^{\pi'}(s)\sum_{a}\pi'(a\mid s)\cdot A^{\pi}(s,a).\end{aligned}$$

The last line uses the normalization $d_{\mu}^{\pi'}(s)=(1-\gamma)\sum_t\gamma^t\cdot\mathbb{P}_{\pi'}(s_t=s)$. Combining with B.2 proves the lemma.

B.4 The Q Form of the Lemma

derive

B.1–B.3 give the advantage form. To reach the inner-product form used in the lecture, note that $V^\pi(s)$ does not depend on the action and both policies sum to one:

$$\begin{aligned}\sum_a \pi'(a \mid s) \cdot A^\pi(s,a) &= \sum_a \pi'(a \mid s) \cdot Q^\pi(s,a) - V^\pi(s) \\ &= \sum_a \pi'(a \mid s) \cdot Q^\pi(s,a) - \sum_a \pi(a \mid s) \cdot Q^\pi(s,a) \\ &= \langle Q^\pi(s,\cdot),\, \pi'(\cdot \mid s) - \pi(\cdot \mid s) \rangle.\end{aligned}$$

The middle line uses $V^\pi(s) = \sum_a \pi(a \mid s) \cdot Q^\pi(s,a)$, which is just the definition of V^π conditioned on the first action.

C.1 The Discarded Term Is Second Order

derive

Continuity alone would only say the two occupancy distributions agree in the limit. The policy gradient needs a rate: each factor below must supply one order of $\|\delta\|$.

Write $c(s,\delta) = \sum_a \pi_{\theta+\delta}(a \mid s) \cdot A^{\pi_\theta}(s,a)$. Advantage centring gives $c(s,0) = 0$ and c is differentiable, so $c(s,\delta) = O(\|\delta\|)$; under standard smoothness of $\theta \mapsto d^{\pi_\theta}$, so is $d^{\pi_{\theta+\delta}}(s) - d^{\pi_\theta}(s)$. Their product,

$$\frac{1}{1-\gamma} \sum_s \{d^{\pi_{\theta+\delta}}(s) - d^{\pi_\theta}(s)\} \cdot c(s,\delta) \;=\; O(\|\delta\|^2),$$

is the gap between the exact identity and the frozen-occupancy surrogate: invisible at first order, which licenses d^{π_θ} in the policy gradient, and dominant once the step is large, which is why PPO must clip.

C.2 The Policy Gradient in Trajectory Form

derive

The theorem can also be proved without the lemma. For $p_\theta(\tau) = \mu(s_0)\prod_t \pi_\theta(a_t \mid s_t)\cdot P(s_{t+1} \mid s_t, a_t)$, only the policy factors depend on θ, so

$$\nabla_\theta \log p_\theta(\tau) = \sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t),$$

and with $\nabla p = p\,\nabla \log p$,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim p_\theta}\Big[G(\tau)\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\Big].$$

Dropping the rewards that precede each action (they are independent of it) turns $G(\tau)$ into $Q^{\pi_\theta}(s_t, a_t)$, and subtracting the baseline $V^{\pi_\theta}(s_t)$ turns it into the advantage. Grouping the time sum by state produces the occupancy distribution.

C.3 Why the Clipped Objective Has Zero Gradient Outside the Band

Take $\widehat{A}_t > 0$. The term is $\widehat{A}_t \cdot \min(\rho_t, 1+\epsilon)$, so

$$\frac{\partial}{\partial \rho_t}[\widehat{A}_t \cdot \min(\rho_t, 1+\epsilon)] = \begin{cases} \widehat{A}_t, & \rho_t < 1+\epsilon, \\ 0, & \rho_t > 1+\epsilon. \end{cases}$$

For $\widehat{A}_t < 0$ the term is $\widehat{A}_t \cdot \max(\rho_t, 1-\epsilon)$ and the derivative vanishes for $\rho_t < 1-\epsilon$. In both cases the gradient is switched off exactly on the side that would keep improving the objective, and left intact on the side that would undo the move. Note the consequence: the sample stops contributing, but other samples in the minibatch may still move the shared parameters, so ρ_t can drift further.